Algoritmos y Estructuras de Datos

OBJETIVOS DE LA ASIGNATURA

Que el alumno conozca las estructuras de datos fundamentales y domine los algoritmos para manipularlas en forma eficiente. Que aprenda a elegir correctamente la estructura de datos y la implementación para obtener el algoritmo más eficiente para un problema dado.

PROGRAMA ANALÍTICO

1. Diseño y análisis de algoritmo. *(2 semanas)*

1.1. Conceptos básicos de algoritmos. Ejemplo: Sincronización de acceso a objetos en cálculo distribuido. Introducción básica a grafos. Planteo del problema mediante grafos. Algoritmo de búsqueda exhaustiva. Generación de las coloraciones. Crecimiento del tiempo de ejecución. Búsqueda exhaustiva mejorada. Algoritmo heurístico. Crecimiento del tiempo de ejecución para el algoritmo ávido. Conclusión del ejemplo.

1.2. Tipos abstractos de datos. Operaciones abstractas y características del TAD conjunto. Interfase del TAD conjunto. Implementación del TAD conjunto. Tiempo de ejecución de un programa. Notación asintótica. Invariancia ante constantes multiplicativas. Invariancia de la tasa de crecimiento ante valores en un conjunto finito de puntos. Transitividad. Regla de la suma. Regla del producto. Funciones típicas utilizadas en la notación asintótica. Equivalencia. La función factorial. Determinación experimental de la tasa de crecimiento. Otros recursos computacionales. Tiempos de ejecución no-polinomiales. Problemas P y NP. Varios parámetros en el problema.

1.4. Conteo de operaciones para el cálculo del tiempo de ejecución. Bloques if. Lazos. Suma de potencias. Llamadas a rutinas. Llamadas recursivas

2. Tipos de datos abstractos fundamentales *(3 semanas)*

2.1. El TAD Lista. Descripción matemática de las listas. Operaciones abstractas sobre listas. Una interfase simple para listas. Funciones que retornan referencias. Ejemplos de uso de la interfase. Implementación de listas por arreglos. Eficiencia de la implementación por arreglos. Implementación mediante celdas enlazadas por punteros. El tipo posición. Celda de encabezamiento. Las posiciones `begin' y `end'. Detalles de implementación. Implementación mediante celdas enlazadas por cursores. Cómo conviven varias celdas en un mismo espacio. Gestión de celdas. Analogía entre punteros y cursores. Tiempos de ejecución de los métodos en las diferentes implementaciones. Interfase STL. Ventajas de la interfase STL. Ejemplo de uso. Uso de templates y clases anidadas. Operadores de incremento prefijo y postfijo. Listas doblemente enlazadas.

2.2. El TAD pila. Ejemplo: Una calculadora RPN con una pila. Operaciones abstractas sobre pilas. Interfase para pila. Implementación de una calculadora RPN. Implementación de pilas mediante listas. La pila como un adaptador. Interfase STL.

2.3. El TAD cola. Ejemplo: Intercalación de vectores ordenados. Ordenamiento por inserción. Tiempo de ejecución. Particularidades al estar las secuencias pares e impares ordenadas. Algoritmo de intercalación con una cola auxiliar. Operaciones abstractas sobre colas. Interfase para cola. Implementación del algoritmo de intercalación de vectores. Tiempo de ejecución.

2.4. El TAD correspondencia. Interfase simple para correspondencias. Implementación de correspondencias mediante contenedores lineales. Implementación mediante contenedores lineales ordenados. Implementación de mediante listas ordenadas. Interfase compatible con STL. Tiempos de ejecución para listas ordenadas. Implementación mediante vectores ordenados. Tiempos de ejecución para vectores ordenados. Definición de una relación de orden.

3. Arboles. *(4 semanas)* Nomenclatura básica de árboles. Altura de un nodo. Profundidad de un nodo. Nivel. Nodos hermanos. Orden de los nodos. Particionamiento del conjunto de nodos. Listado de los nodos de un árbol. Orden previo. Orden posterior. Orden posterior y la notación polaca invertida. Notación Lisp para árboles. Reconstrucción del árbol a partir de sus órdenes.

3.3. Operaciones con árboles. Algoritmos para listar nodos. Inserción en árboles. Algoritmo para copiar árboles. Supresión en árboles. Operaciones básicas sobre el tipo árbol.

3.4. Interfase básica para árboles. Listados en orden previo y posterior y notación Lisp. Funciones auxiliares para recursión y sobrecarga de funciones. Algoritmos de copia. Algoritmo de poda. Implementación de la interfase básica por punteros.

3.5.1. El tipo iterator. Las clases `cell' e `iterator'. La clase tree. Interfase avanzada. Ejemplo de uso de la interfase avanzada.

3.7. Tiempos de ejecución

3.8 Arboles binarios. Listados en orden simétrico. Notación Lisp. Árbol binario lleno. Operaciones básicas sobre árboles binarios. Interfases e implementaciones. Interfase básica. Predicados de igualdad y espejo. Hacer espejo "in place". Implementación con celdas enlazadas por punteros . El algoritmo apply y principios de programación funcional. . Implementación de la interfase avanzada.

3.9 Arboles de Huffman. Condición de prefijos. Representación de códigos como árboles de Huffman. Códigos redundantes. Tabla de códigos óptima. Algoritmo de búsqueda exhaustiva. Generación de los árboles. Un toque de programación funcional. El algoritmo de combinación. Función auxiliar que calcula la longitud media . El algoritmo de Huffman. Implementación del algoritmo. Ejemplo: Un programa de compresión de archivos.

4. Conjuntos. *(3 semanas)* Introducción a los conjuntos. Notación de conjuntos. Interfase básica para conjuntos. Análisis de flujo de datos.

4.2. Implementación por vectores de bits. Conjuntos universales que no son rangos contiguos de enteros. Descripción del código.

4.3. Implementación con listas. Similaridad entre los TAD conjunto y correspondencia. Algoritmo lineal para las operaciones binarias. Descripción de la implementación. Tiempos de ejecución.

4.4. Interfase avanzada para conjuntos.

4.5. El diccionario. La estructura tabla de dispersión. Tablas de dispersión abiertas. Detalles de implementación. Tiempos de ejecución. Funciones de dispersión. Tablas de dispersión cerradas. Costo de la inserción exitosa. Costo de la inserción no exitosa. Costo de la búsqueda. Supresión de elementos. Costo de las funciones cuando hay supresión. Reinserción de la tabla. Costo de las operaciones con supresión. Estrategias de redispersión. Detalles de implementación.

4.6. Conjuntos con árboles binarios de búsqueda. Representación como lista ordenada de los valores. Verificar la condición de ABB. Mínimo y máximo. Buscar un elemento. Costo de mínimo. Operación de inserción. Operación de borrado. Recorrido en el árbol. Operaciones binarias. Detalles de implementación. Tiempos de ejecución. Balanceo del árbol.

5. Ordenamiento. *(2 semana)* Introducción. Relaciones de orden débiles. Signatura de las relaciones de orden. Predicados binarios. Relaciones de orden inducidas por composición. Estabilidad. Primeras estimaciones de eficiencia. Algoritmos de ordenamiento en las STL.

5.2. Métodos de ordenamiento lentos. El método de la burbuja. El método de inserción. El método de selección. Comparación de los métodos lentos. Estabilidad.

5.3. Ordenamiento indirecto. Minimizar la llamada a funciones.

5.4. El método de ordenamiento rápido, quick-sort. Tiempo de ejecución. Casos extremos. Elección del pivote. Tiempo de ejecución. Caso promedio. Dispersión de los tiempos de ejecución. Elección aleatoria del pivote. El algoritmo de partición . Tiempo de ejecución del algoritmo de particionamiento. Búsqueda del pivote por la mediana. Implementación de quick-sort. Estabilidad. El algoritmo de intercambio (swap). Tiempo de ejecución del quick-sort estable.

5.5. Ordenamiento por montículos. El montículo. Propiedades. Inserción. Costo de la inserción. Eliminar el mínimo. Costo de re-heap. Implementación in-place. El procedimiento make-heap. Implementación. Propiedades de la clasificación por montículo.

5.6. Ordenamiento por fusión. Implementación. Estabilidad. Versión estable de split. Merge-sort para vectores. Clasificación externa.

5.7. Comparación de algunas implementaciones de algoritmos de ordenamiento.

6. Técnicas de Análisis de Algoritmos. *(1 semanas)* Eficiencia de los algoritmos. Análisis de programas recursivos. Resolución de ecuaciones de recurrencia. Solución general para una clase grande de recurrencias.

7. Técnicas de Diseño de Algoritmos. *(1 semanas)* Algoritmos dividir para vencer. Programación dinámica. Algoritmos ávidos (greedy). Método de retroceso (Backtracking). Algoritmos de búsqueda local.

LISTADO Y DESCRIPCIÓN DE ACTIVIDADES PRÁCTICAS

- Guía 1: Diseño y análisis de algoritmos. Tipos de datos abstractos fundamentales.
- Guía 2: Tipos de datos abstractos fundamentales.
- Guía 3: Arboles I.
- Guía 4: Arboles II.
- Guia 5: Operaciones básicas con conjuntos I.
- Guia 6: Operaciones básicas con conjuntos II.
- Guia 7: Clasificación.
- Guia 8: Tecnicas de diseño de Algoritmos.

BIBLIOGRAFÍA

- **Apuntes de la cátedra: http://www.cimec.org.ar/aed**
- **Aho, Hopcroft y Ullman, Estructura de Datos Y Algoritmos, Ed. Addison-Wesley (1988).**
- **Tenenbaum y Augenstein, Estructura de Datos en Pascal, Ed. Prentice-Hall (1983).**
- **Weiss, Estructuras de Datos y Algoritmos, Ed. Addison Wesley (1995).**
- **Wirth, Algoritmos y Estructura de Datos, Ed. Prentice Hall (1987).**